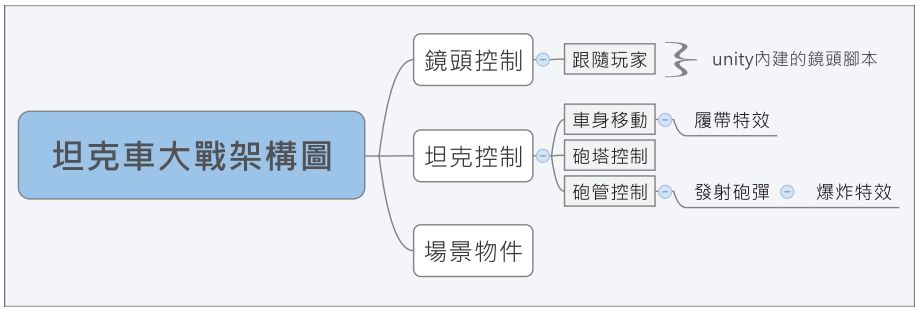
6.1 | 前言

1. 本章範例將介紹坦克車大作戰遊戲製作，操作方式是屬於第三人稱遊戲。主角為一台坦克車，它可以分成三個部份：車身、砲塔及砲管。我們可以先打開完成後的執行檔玩玩看。

- 利用鍵盤的方向鍵，可以位移及旋轉坦克車。
- 將滑鼠水平移動，可以改變砲塔的方向。
- 轉動滑鼠的中鍵滾輪，可以控制砲管上下角度，按下滑鼠左鍵可以發射砲彈。
- 若要離開遊戲，按下 Alt + F4 即可。

2. 本範例將會逐一介紹幾個常用技巧：第三人稱角色控制方式、鏡頭跟隨主角、坦車克履帶製作、利用鏡頭加燈光、發射砲彈及如何避免砲彈發射時炸到自己…等。

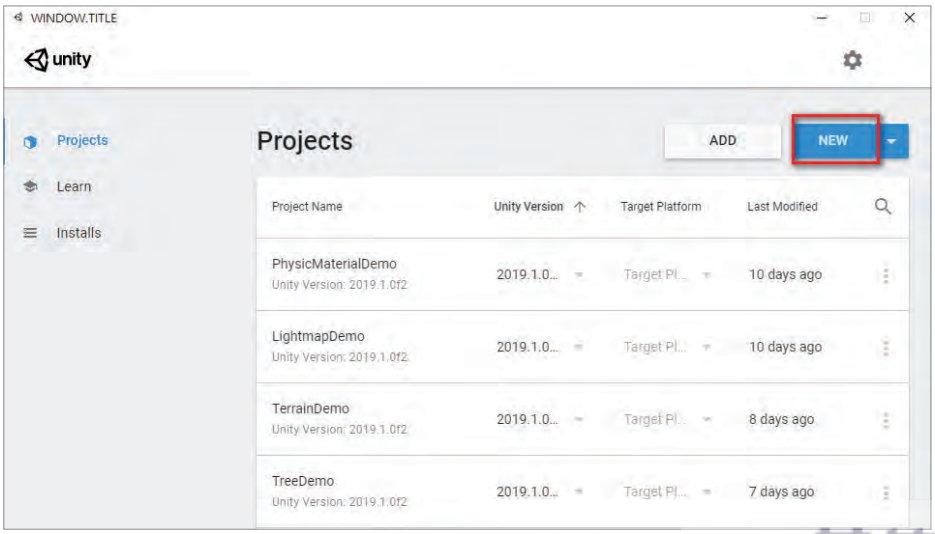
3. 遊戲架構概念圖



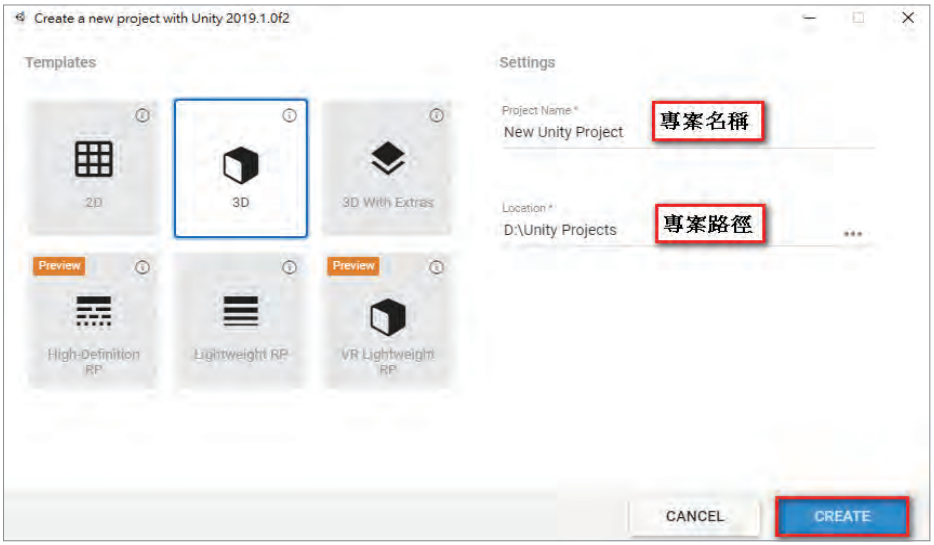
6.2 | 坦克移動

本節將介紹如何將模型匯入場景、打光、控制坦克之方式、第三人稱鏡頭的操作…等技巧。另外也將逐步探討語法的應用，運用編寫腳本，使坦克能夠移動、旋轉。

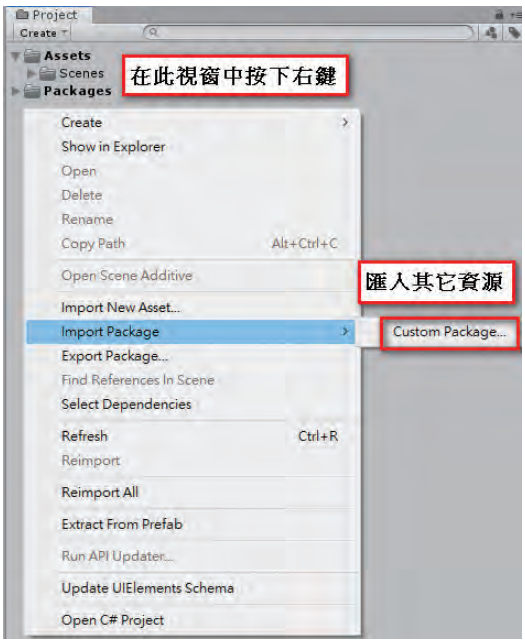
1. **新建專案。**開啟 Unity Hub 後，選擇 New 建立新專案。

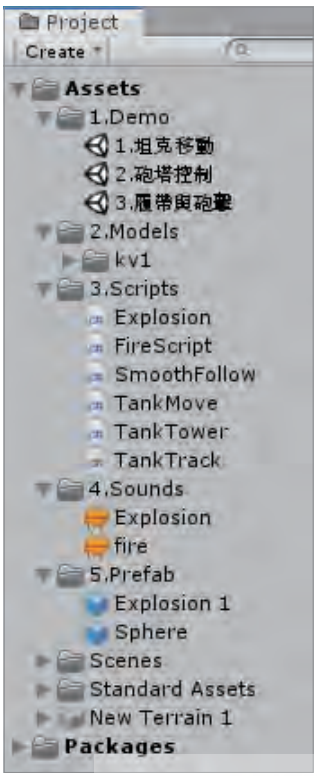


☛ 設定好名稱與路徑後，按下 CREATE 建立專案。

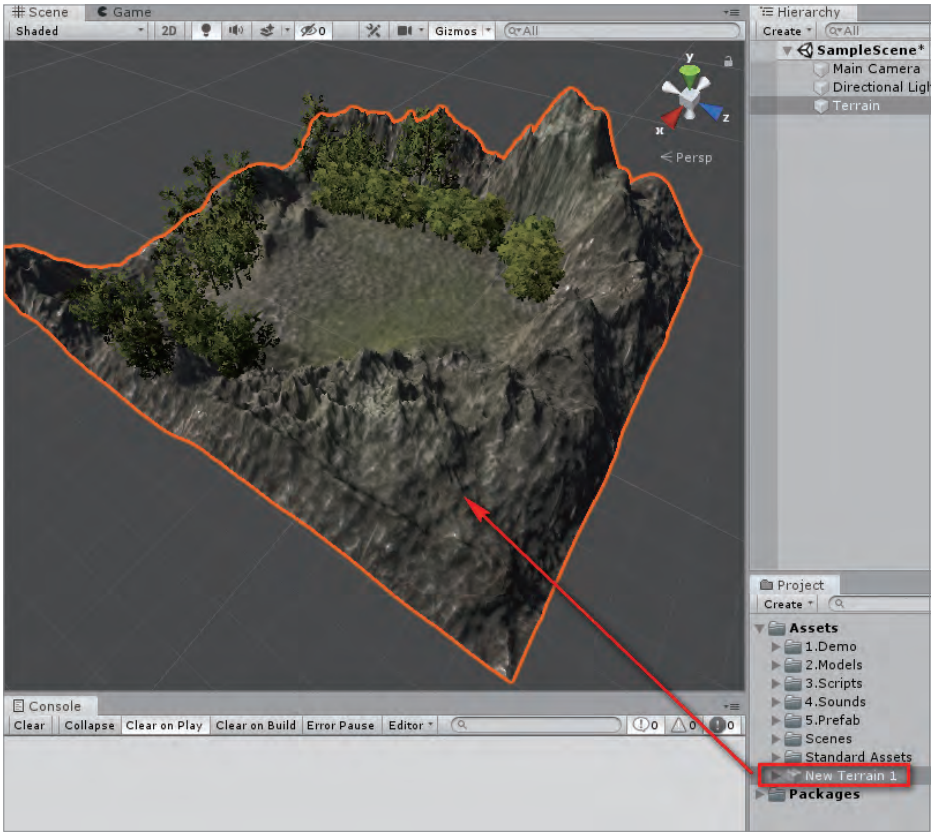


2. **匯入資源包**。從 Project 視窗中按右鍵 → Import Package → Custom Package 將範例的資源包 (Demo6.unitypackage) 匯入。





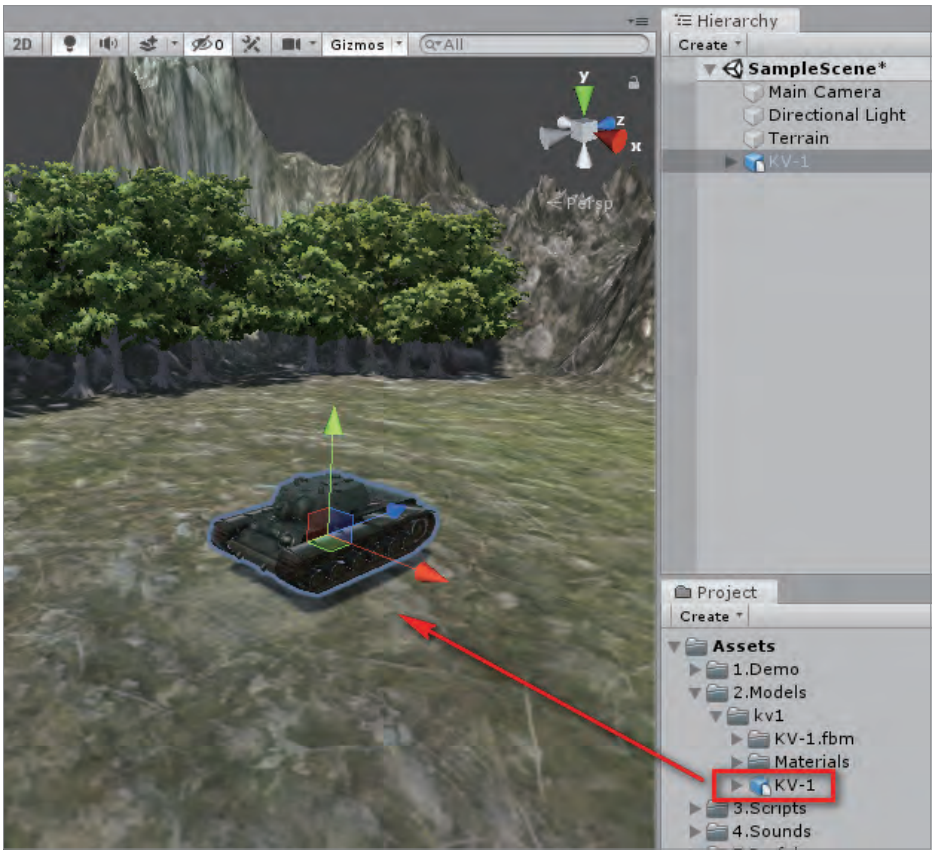
3. **組裝場景**。將 Project 視窗中的 New Terrain 1 拖到 Scene 視窗。



這是已經編輯好的地形。

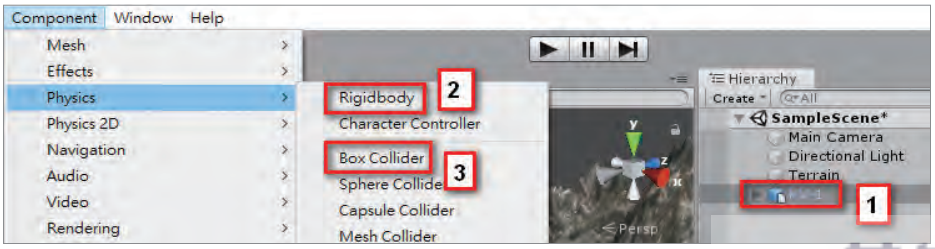
選擇 Hierarchy 視窗中的地形，先將它的位置歸零，這樣對之後加入的物件會比較好調整位置，即 (x,y,z)=(0,0,0)，或者按下 transform 的右側的倒三角形，即可出現下拉式功能表，選取 Reset Position。

4. **組裝坦克**。開始製作可以操控的坦克，從 Project 視窗中 → 2.Models → KV-1 資料夾裡，將 KV-1 模型拉到 Scene 視窗。



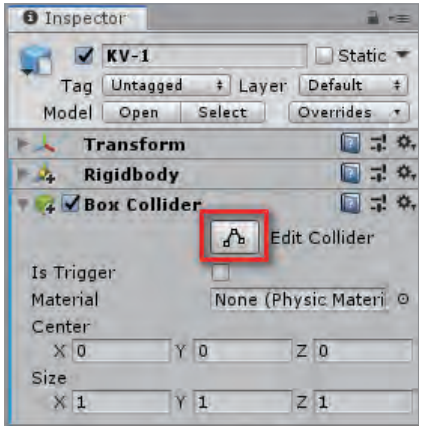
必須確認場景中的模型需高於地面，以免遊戲在執行時，坦克掉落到地面之下。

選擇 Hierarchy 視窗中的坦克 KV-1，從 Component → Physics 裡附加 Rigidbody 與 Box Collider 這兩個組件給它。

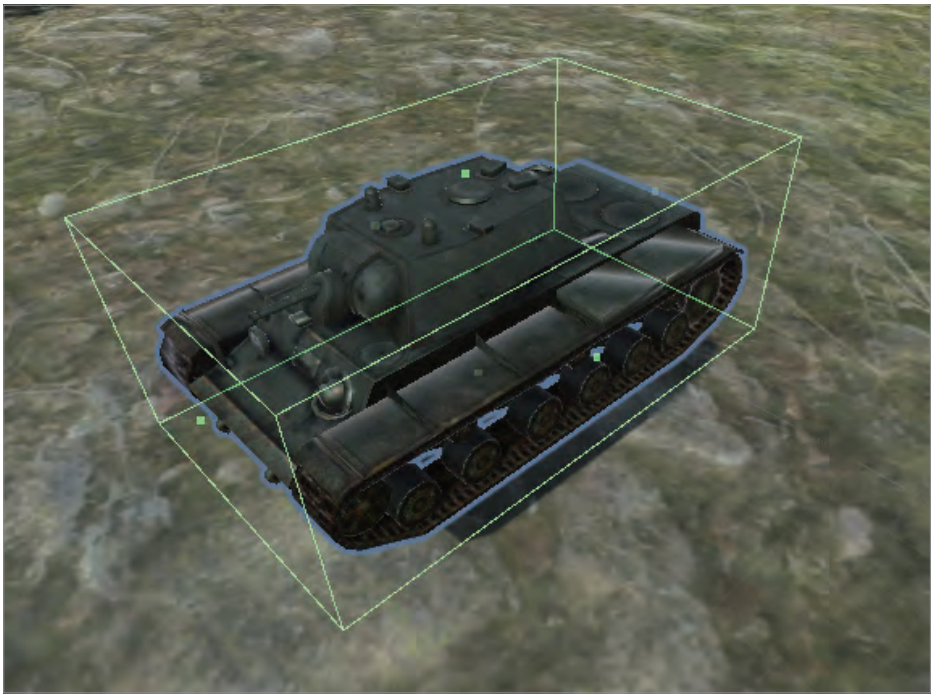


- Rigidbody 組件可以使坦克有物理效果，當按下 Play 鍵  ，我們就會發現坦克因重力而掉到地面上。
- Box Collider 組件可以使坦克有碰撞框。

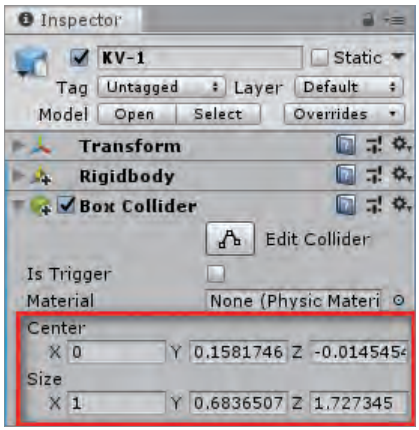
☛ 選擇坦克，在 Inspector 視窗中的 Box Collider 組件，按下 Edit Collider。



☛ 此時在 Scene 視窗中就會出現控制點，把碰撞框調整到符合坦克的大小。

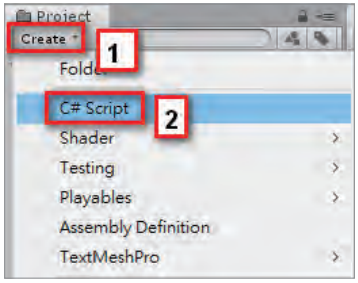


除此之外也可以在 Inspector 視窗的 scale 做調整。

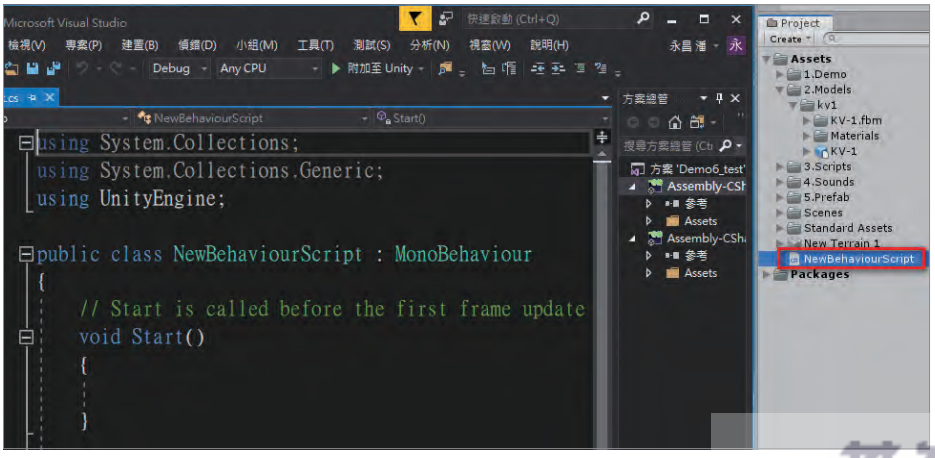


5. **光源。**新場景會預設一個 Directional light(平行光)，通常我們用它來模擬太陽光 (平行光不受位置的影響，只有旋轉角度會影響照明)。

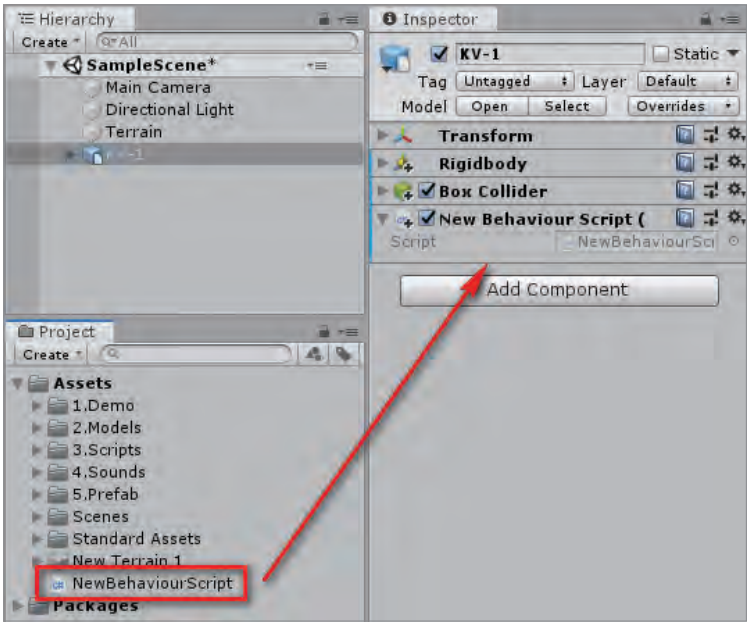
6. **編寫能夠操控坦克的腳本。**從 Project 視窗旁按下 Create，選擇 C# 建立一個新腳本。



從 Project 視窗中對新腳本點擊兩下，來開啟編輯視窗。



☛ 將剛剛建立的腳本拖給坦克，我們才能夠邊寫腳本邊測試。



7. **坦克操控設定。**在之前的基礎語法章節中有提到讓物體移動、旋轉的語法，現在就應用它們來操控坦克。

☛ 在腳本編輯視窗中找到 Update，在大括號裡面寫入：

transform.Translate(0,0,1);

與

transform.Rotate(0,1,0);

```
public class NewBehaviourScript : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(0, 0, 1);
        transform.Rotate(0, 1, 0);
    }
}
```

- transform.Translate(x,y,z)，可以讓物體朝指定的軸向來移動。
- transform.Rotate(x,y,z)，可以讓物體朝指定的軸向來旋轉。

- 當按下 Play 鍵，會發現坦克很快的離開 Scene 視窗，因為現在只是讓坦克按照指定的軸向來移動、旋轉，我們還沒加入控制。

- 加入鍵盤的按鍵控制，在移動、旋轉的語法之前，寫入

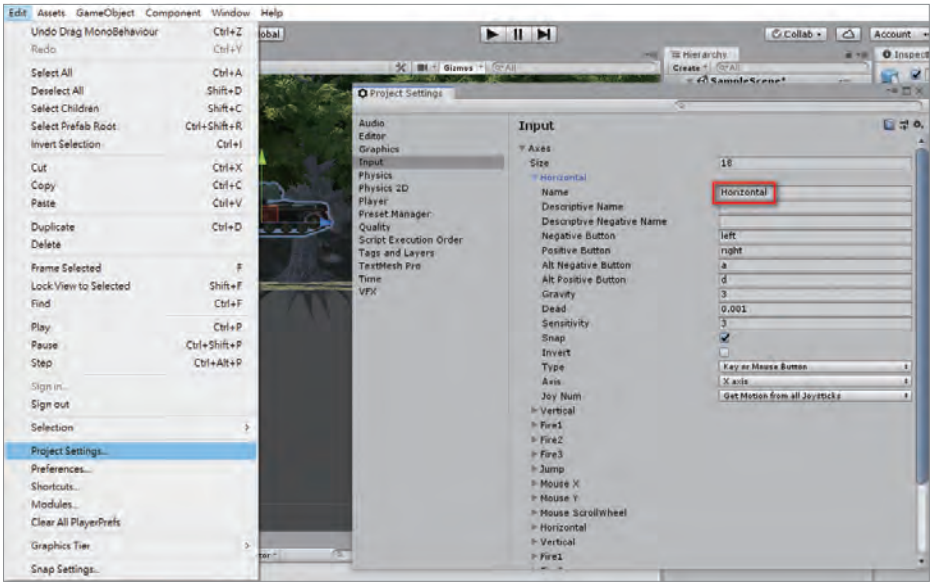
var h = Input.GetAxis("Horizontal");

與

var v = Input.GetAxis("Vertical");

```
// Update is called once per frame
void Update()
{
    var h = Input.GetAxis("Horizontal");
    var v = Input.GetAxis("Vertical");

    transform.Translate(0, 0, 1);
    transform.Rotate(0, 1, 0);
}
```

- Input.GetAxis 能夠讀取 Edit → Project Settings → Input 裡設置的按鍵軸向，在括弧內必須填寫按鍵的名稱。



- Horizontal 與 Vertical 分別是水平、垂直，它們代表鍵盤上的 WASD 與上下左右鍵。

- 將移動、旋轉語法更改為 transform.Translate(0,0,1 * v); 與 transform.Rotate(0,1 * h,0);

```
transform.Translate(0, 0, 1 * v);
transform.Rotate(0, 1 * h, 0);
```

- 當按下 Play 鍵，我們會發現坦克能依照鍵盤來控制了。

- 在測試的時候，我們會發現當按下 W 鍵 (向前)，但坦克是向後跑，那是因為模型的方向與軸向相反。
  - 箭頭的顏色代表 (X) 紅、(Y) 綠、(Z) 藍。
  - 只要修改語法，即可讓坦克移動的方向正確，將 1 * v 改為 1 * -v。

```
transform.Translate(0, 0, 1 * -v);
transform.Rotate(0, 1 * h, 0);
```

- 在測試時會發現坦克的移動、旋轉速度太快，如果要一直修改語法到滿意為止，會很麻煩，所以我們宣告移動、旋轉的變數，宣告後的變數能在 Inspector 視窗中調整，這樣會方便許多。

- 在所有 void 之外的空白處寫入

  public float mSpeed = 1;

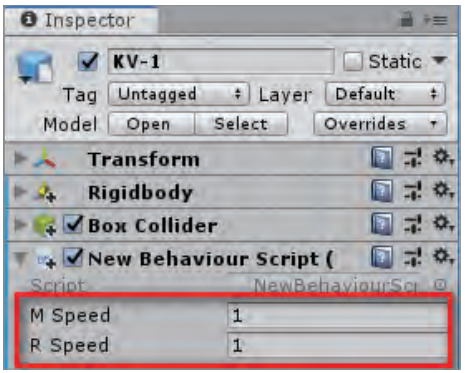
  與

  public float rSpeed = 1;

  - 變數的名稱可以隨意取，但建議命名為有意義的名稱。
  - float 表示為浮點數。
  - 關鍵字為 public 時，在 Inspector 視窗中才會出現它們的參數。

```
public class NewBehaviourScript : MonoBehaviour
{
    public float mSpeed = 1;
    public float rSpeed = 1;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        var h = Input.GetAxis("Horizontal");
        var v = Input.GetAxis("Vertical");

        transform.Translate(0, 0, 1 * -v);
        transform.Rotate(0, 1 * h, 0);
    }
}
```

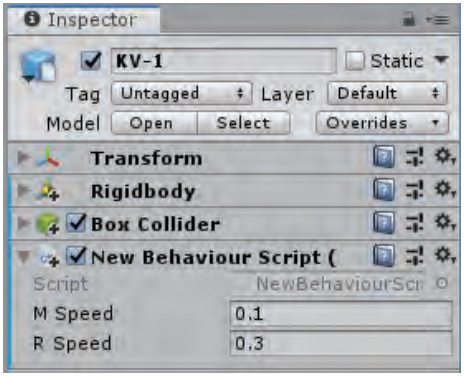
- 若在 void 內宣告的變數，只能在該 void 使用，且 Inspector 視窗中不會出現它們的參數。



- 現在還無法修改參數來改變坦克的速度，必須先將原本的語法分別改為 mSpeed * -v 與 rSpeed * h，更改後就能夠透過參數調整來改變坦克的速度。

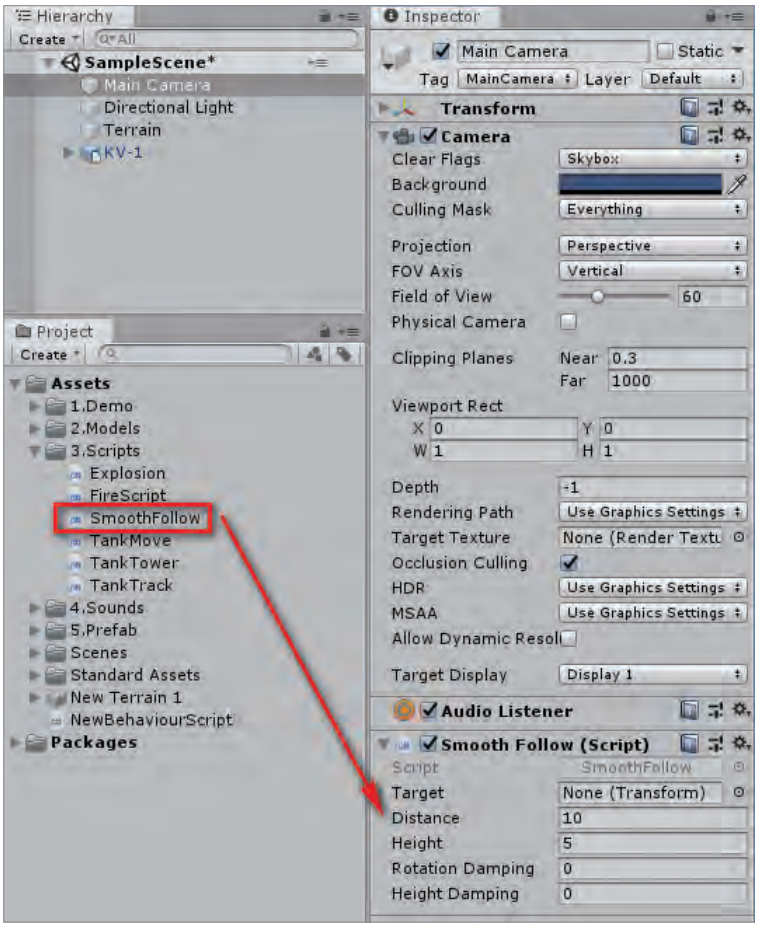
```
transform.Translate(0, 0, mSpeed * -v);
transform.Rotate(0, rSpeed * h, 0);
```

- 在調整參數的時候，可以先在執行中 (按下 Play 鍵) 調整，但是在執行的時候，任何的參數調整都不會記錄下來，所以要將調整好的數值自行記下來，等結束執行後 (按下 stop 按鈕) 再填入。本範例建議把 M Speed 設為 0.1、R Speed 設為 0.3，讓坦克的運行速度慢一些。



8. **鏡頭跟隨。**現在已經能夠透過鍵盤來操控坦克了，但是在 Game 視窗卻無法看見坦克，因為 Camera 並沒有跟隨坦克移動，所以我們要加入腳本來控制他，使之能夠跟隨坦克移動。

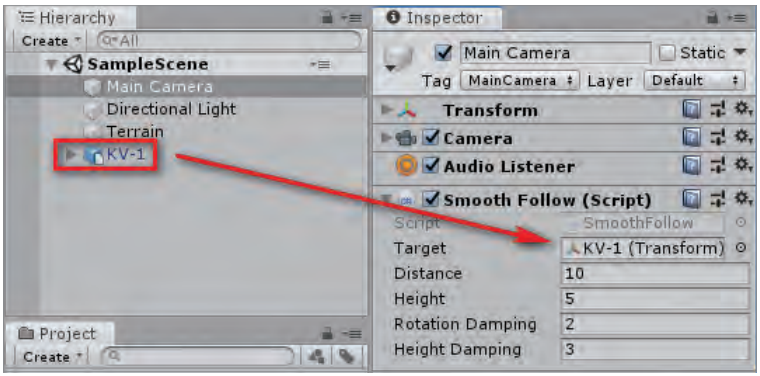
本範例已經準備好鏡頭跟隨的腳本了，所以我們直接從 Hierarchy 視窗中選擇 Main Camera，將 SmoothFollow 腳本附加給它。



SmoothFollow 腳本的參數介紹

Target	Camera 跟隨的目標	KV-1
Distance	Camera 跟目標保持的距離	10
Height	Camera 跟目標保持的高度	5
Rotation Damping	目標旋轉時 Camera 跟隨的平滑度	2
Height Damping	目標高度改變時 Camera 跟隨的平滑度	3

☛ 我們先試著將坦克拉入 Target 參數中，並將 Rotation Damping 參數設為 2、Height Damping 參數設為 3。

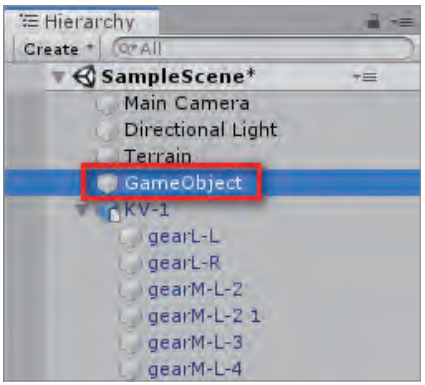


按下 Play 鍵後，我們會發現 Game 視窗出現坦克的蹤影，當移動坦克時，Camera 也會跟隨。

9. **調整坦克方向。**眼尖的人或許會發現 Game 視窗中的坦克是一直朝自己的方向來移動，因為模型的方向與軸向相反，而且從 Scene 視窗中可以發現 Camera 的視角會關注著坦克的軸向。

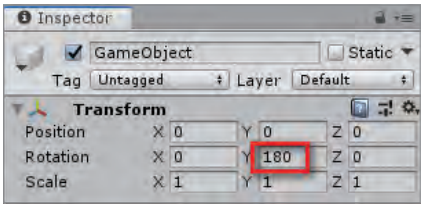
為了解決這個問題，我們選擇坦克 (KV-1)，並從 GameObject → Create Empty Child，建立一個空物件來修正軸向。

- 使用 Create Empty 會直接生在外面，並不會自動跑到選擇物件的子層級。

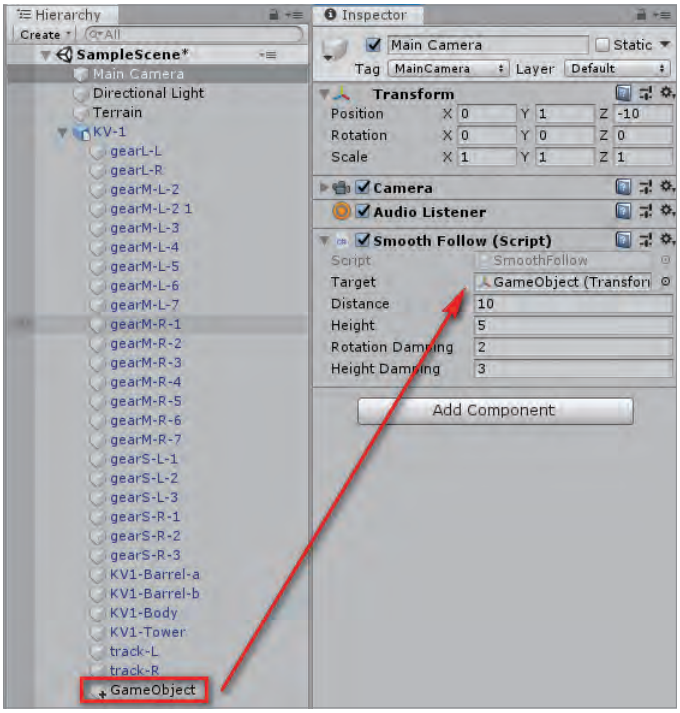


☛ 將 GameObject 的 Y 軸旋轉 180 度。

- 此時 GameObject 的軸向會與模型的方向相同。



☛ 調整好後，將 GameObject 拉到 Main Camera 的 SmoothFollow 腳本裡的 Target 參數中。



修改好後，我們再按下 Play 鍵，就會發現 Game 視窗中的坦克移動方向是正確的。

10. 現在大家已經學會如何透過腳本來控制坦克了，希望大家可以多練習，下一節將增加腳本的內容，控制砲塔旋轉與砲管仰角。

6.3 | 砲塔控制

本節將介紹如何左右移動滑鼠來控制砲塔的旋轉量，與如何使用滑鼠的中鍵滾輪來控制砲管的上下角度。

1. **開啟場景。**您可以接續上一個章節的場景檔案，或者也可以開啟 Project 視窗中的 1.Demo → 1. 坦克移動，以方便本節的進度繼續進行。

2. **調整坦克的層級關係。**在編寫語法之前，要先了解這個坦克的模型有哪些子物件。